Parallel Space-Time Likelihood Optimization for Air Pollution Prediction on Large-Scale Systems

Mary Lai O. Salvaña
Extreme Computing Research Center, King Abdullah University of Science and Technology
Thuwal, Saudi Arabia
marylai.salvana@kaust.edu.sa

Sameh Abdulah
Extreme Computing Research Center, King Abdullah University of Science and Technology
Thuwal, Saudi Arabia
sameh.abdulah@kaust.edu.sa

Hatem Ltaief
Extreme Computing Research Center, King Abdullah University of Science and Technology
Thuwal, Saudi Arabia
hatem.ltaief@kaust.edu.sa

Ying Sun
Extreme Computing Research Center, King Abdullah University of Science and Technology
Thuwal, Saudi Arabia
ying.sun@kaust.edu.sa

Marc G. Genton
Extreme Computing Research Center, King Abdullah University of Science and Technology
Thuwal, Saudi Arabia
marc.genton@kaust.edu.sa

David E. Keyes
Extreme Computing Research Center, King Abdullah University of Science and Technology
Thuwal, Saudi Arabia
david.keyes@kaust.edu.sa

ABSTRACT

Gaussian geostatistical space-time modeling is an effective tool for performing statistical inference of field data evolving in space and time, generalizing spatial modeling alone at the cost of the greater complexity of operations and storage, and pushing geostatistical modeling even further into the arms of high-performance computing. It makes inferences for missing data by leveraging space-time measurements of one or more fields. We propose a high-performance implementation of a widely applied space-time model for large-scale systems using a two-level parallelization technique. At the inner level, we rely on state-of-the-art dense linear algebra libraries and parallel runtime systems to perform complex matrix operations required to evaluate the maximum likelihood estimation (MLE). At the outer level, we parallelize the optimization process using a distributed implementation of the particle swarm optimization (PSO) algorithm. At this level, parallelization is accomplished using MPI sub-communicators, such that the nodes in each sub-communicator perform a single MLE iteration at a time. To evaluate the effectiveness of the proposed methodology, we assess the accuracy of the newly implemented space-time model on a set of large-scale synthetic space-time datasets. Moreover, we use the proposed implementation to model two air pollution datasets from the Middle East and US regions with 550 spatial locations ×730 time slots and 945 spatial locations ×500 time slots, respectively. The evaluation shows that the proposed approach satisfies high prediction accuracy on both synthetic datasets and real particulate matter (PM) datasets in the context of the air pollution problem. We achieve up to 757.16 TFLOPS/s using 1024 nodes (75% of the peak performance) using 490K geospatial locations on Shaheen-II Cray XC40 system.

PASC '22, June 27–29, 2022, Basel, Switzerland

ACM ISBN 978-1-4503-9410-9/22/06.
https://doi.org/10.1145/3539781.3539800

CCS CONCEPTS

• **Applied computing** → **Environmental sciences**.

KEYWORDS

Geostatistics, High-Performance Computing, Space-Time Modeling.

ACM Reference Format:
Mary Lai O. Salvaña, Sameh Abdulah, Hatem Ltaief, Ying Sun, Marc G. Genton, and David E. Keyes. 2022. Parallel Space-Time Likelihood Optimization for Air Pollution Prediction on Large-Scale Systems. In *Platform for Advanced Scientific Computing Conference (PASC '22), June 27–29, 2022, Basel, Switzerland*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3539781.3539800

1 INTRODUCTION

Gaussian geostatistical space-time modeling has emerged as an essential complement to physics-based modeling of space-time processes through, e.g., discretized partial differential equations, because it is widely applicable and straightforward to use with sufficient data. As observational data become more available, modelers have turned to Gaussian geostatistics for its inference and prediction capabilities. Gaussian geostatistical models embed correlations arising from causality and are adequate for many phenomena. The challenges to Gaussian geostatistical modeling are the cubic growth of computational complexity and the quadratic increase of storage complexity in the number of observations correlated through the Gaussian log-likelihood evaluation process. This complexity is linked to the Cholesky factorization operation required to compute the inverse of the application-associated covariance matrix. Fitting the models involves an iterative procedure requiring a cubic complexity to evaluate the log-likelihood at each iteration. As shown herein, these are addressable through multi-level parallelism using asynchronous executions of computational tasks.

The grand challenge in large-scale Gaussian geostatistical modeling lies in the inversion of $\Sigma(\boldsymbol{\theta})$. The MLE operation involves an iterative optimization process to evaluate the log-likelihood function in order to find the optimum set of parameters that obtains the

global maximum likelihood value. We adopt a parallel implementation of the particle swarm optimization (PSO) algorithm [37], which belongs to the family of evolutionary computing algorithms. The PSO algorithm depends on the interaction between independent particles to scan the search space of the problem to find the globally optimum solutions [40]. It has several advantages, including its low computational cost and amenability to parallelization on large-scale systems. The implementation in [37] incorporates the pattern search method to the PSO algorithm to generate a more aggressive version of the optimization process that allows more accurate estimations of the required optimum solutions.

In this work, we use a two-level parallelization technique to perform the MLE operation in space-time dimensions. The inner level involves matrix operations of the log-likelihood function using parallel task-based linear solvers. The outer level involves the parallel PSO (PPSO) algorithm to distribute the optimization process between different numbers of nodes with the aid of the Message Passing Interface (MPI) API. These techniques are tested on air pollution datasets. The dangers of high levels of pollution, i.e., the mixture of solid particles and liquid droplets suspended in the air known as particulate matter (PM), have been revealed in a broad range of studies. High concentrations of PM can cause cardiovascular, respiratory, and Parkinson's diseases and decrease life expectancy [15, 21, 22, 38]. PM also has devastating effects on the environment, including disturbances in nutrient cycling [14], cloud formation, solar radiation, the occurrence of acid rain, and acceleration of climate change [10]. Through its long-range transport patterns, PM can travel to other locations and spread infectious diseases [16]. Understanding and predicting the evolution of PM is key to mitigating its effects.

The eight-fold contributions of the paper are as follows: 1) we present a high-performance implementation of a flexible Gaussian geostatistical space-time model on large-scale systems involving the log-likelihood function; 2) we provide visualizations of the space-time random fields generated by the high-performance implementation; 3) we incorporate the PPSO algorithm to the MLE operation to utilize the execution performance on distributed environments; 4) we exploit the MPI communicators paradigm to create a set of independent executions for the log-likelihood function that allows parallel execution of the MLE operation while relying on task-based parallel linear solvers to perform the log-likelihood matrix operations; 5) we can achieve up to 757.16 TFLOPS/s using 1024 nodes on the Shaheen-II Cray XC40 system (75% of the peak performance); 6) we show that using the PPSO algorithm, all the parameters of the space-time model can be effectively estimated; 7) we illustrate the benefits of using the newly implemented space-time model over a simpler space-time model with fewer parameters via large scale space-time experiments; 8) we apply our new implementation on two air pollution datasets from the Middle East and US regions. The evaluation shows the advantage of the proposed approach in satisfying high prediction accuracy in such applications.

The remainder of the paper is structured as follows. Section 2 gives the general background on Gaussian geostatistical space-time models and their associated covariance functions. We also recall the PSO algorithm for parallel optimizations. Section 3 highlights related works and positions our research contributions. Section 4 contains a detailed description of the proposed implementation on distributed-memory systems. Section 5 reports the performance results in terms of accuracy/time-to-solution, and we conclude in Section 6.

2 BACKGROUND OF THE STUDY

In this section, we give an overview of Gaussian geostatistical space-time models with their covariance functions and describe the PSO algorithm and its parallel implementation.

2.1 Gaussian Space-Time Modeling/Prediction

Gaussian geostatistical space-time models offer a way to model space-time data by considering them as realizations of a space-time random field. Denote the measurement obtained at space-time location $(\mathbf{s}, t) \in \mathbb{R}^d \times \mathbb{R}, d \geq 1$, as $Z(\mathbf{s}, t)$, such that the elements in the vector of measurements $\mathbf{Z} = \{Z(\mathbf{s}_1, t_1), Z(\mathbf{s}_2, t_2), \ldots, Z(\mathbf{s}_n, t_n)\}^\top$, where $n \in \mathbb{Z}^+$ is the number of space-time locations, come from the distribution of $Z(\mathbf{s}, t)$ with mean $\mathrm{E}\{Z(\mathbf{s}, t)\} = \mu(\mathbf{s}, t)$ and covariance $\mathrm{cov}\{Z(\mathbf{s}_1, t_1), Z(\mathbf{s}_2, t_2)\} = C(\mathbf{s}_1, \mathbf{s}_2; t_1, t_2|\boldsymbol{\theta})$. Here $\mu(\mathbf{s}, t)$ is the mean function that models the deterministic part of the space-time random field and $C(\mathbf{s}_1, \mathbf{s}_2; t_1, t_2|\boldsymbol{\theta})$ is a space-time covariance function, parameterized by $\boldsymbol{\theta} \in \mathbb{R}^q, q \geq 1$, that describes the strength of dependence of the measurements at any two space-time locations $(\mathbf{s}_1, t_1)$ and $(\mathbf{s}_2, t_2)$. A space-time random field is second-order stationary in space and time when the covariance depends only on the space-time lag $(\mathbf{h}, u)$. That is, suppose $Z(\mathbf{s}, t)$ is a second-order stationary space-time random field. Then its covariance is given by $C(\mathbf{s}_1, \mathbf{s}_2; t_1, t_2|\boldsymbol{\theta}) = C(\mathbf{h}, u|\boldsymbol{\theta})$, where $\mathbf{h} = \mathbf{s}_1 - \mathbf{s}_2$ and $u = t_1 - t_2$. The parametric space-time covariance function C can be chosen among many established models in the Gaussian geostatistical literature [7, 13]. A typical workflow in Gaussian geostatistical space-time modeling involves estimating the parameter vector $\boldsymbol{\theta}$ given space-time measurements $\mathbf{Z}$. This is often done via the maximum likelihood estimation (MLE). In MLE, the Gaussian log-likelihood function

$$l(\boldsymbol{\theta}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma(\boldsymbol{\theta})| - \frac{1}{2}\mathbf{Z}^\top\Sigma(\boldsymbol{\theta})^{-1}\mathbf{Z}, \tag{1}$$

is maximized with respect to $\boldsymbol{\theta}$. Here $\Sigma(\boldsymbol{\theta}) = \{C(\mathbf{s}_i, \mathbf{s}_j; t_i, t_j|\boldsymbol{\theta})\}_{i,j=1}^n$ is the $n \times n$ covariance matrix of $\mathbf{Z}$ and $|\Sigma(\boldsymbol{\theta})|$ is the determinant of $\Sigma(\boldsymbol{\theta})$. The covariance matrix $\Sigma(\boldsymbol{\theta})$ represents the dependence between different pairs of locations based on the chosen space-time covariance function.

Once the maximum likelihood estimates, denoted $\hat{\boldsymbol{\theta}}$, are obtained, prediction, also known as kriging, can be done by recognizing that the vector of known space-time measurements $\mathbf{Z}_1$ and the vector of missing space-time measurements $\mathbf{Z}_2$ are jointly Gaussian, i.e.,

$$\begin{bmatrix}\mathbf{Z}_1\\ \mathbf{Z}_2\end{bmatrix} \sim \mathcal{N}_{n_1+n_2}\left(\begin{bmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{bmatrix}, \begin{bmatrix}\Sigma_{11}(\boldsymbol{\theta}) & \Sigma_{12}(\boldsymbol{\theta})\\ \Sigma_{21}(\boldsymbol{\theta}) & \Sigma_{22}(\boldsymbol{\theta})\end{bmatrix}\right), \tag{2}$$

where n_1 and n_2 are the number of given and missing space-time locations, respectively, $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ are the mean vectors of $\mathbf{Z}_1$ and $\mathbf{Z}_2$, respectively, Σ_{11} and Σ_{22} are the covariance matrices of $\mathbf{Z}_1$ and $\mathbf{Z}_2$, respectively, and $\Sigma_{12} = \Sigma_{21}^\top$ is the cross-covariance matrix of $\mathbf{Z}_1$ and $\mathbf{Z}_2$. The conditional distribution of the missing space-time

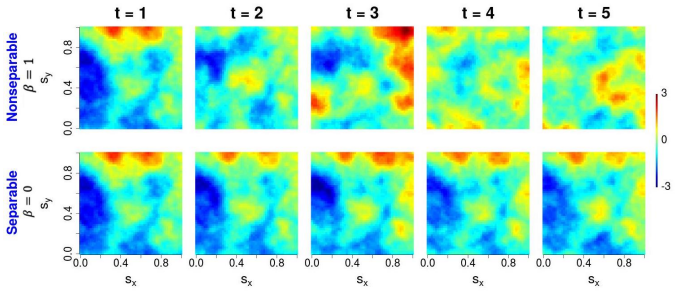


Figure 1: Simulated space-time realizations from the model in (6) with different values of the space-time interaction parameter, β. The nonseparable and separable models correspond to $\beta = 1$ and $\beta = 0$, respectively.

measurements $\mathbf{Z}_2$ is:

$$\mathbf{Z}_2|\mathbf{Z}_1 \sim \mathcal{N}_{n_2}\{\boldsymbol{\mu}_2 + \Sigma_{21}(\boldsymbol{\theta})\Sigma_{11}^{-1}(\boldsymbol{\theta})(\mathbf{Z}_1 - \boldsymbol{\mu}_1), \\ \Sigma_{22}(\boldsymbol{\theta}) - \Sigma_{21}(\boldsymbol{\theta})\Sigma_{11}^{-1}(\boldsymbol{\theta})\Sigma_{12}(\boldsymbol{\theta})\}. \quad (3)$$

Under the zero-mean assumption, the best linear unbiased predictor of $\mathbf{Z}_2$ is:

$$\hat{\mathbf{Z}}_2 = \Sigma_{21}(\hat{\boldsymbol{\theta}})\Sigma_{11}^{-1}(\hat{\boldsymbol{\theta}})\mathbf{Z}_1, \quad (4)$$

with prediction uncertainty

$$\hat{\mathbf{V}}_2 = \text{diag}\{\Sigma_{22}(\hat{\boldsymbol{\theta}}) - \Sigma_{21}(\hat{\boldsymbol{\theta}})\Sigma_{11}^{-1}(\hat{\boldsymbol{\theta}})\Sigma_{12}(\hat{\boldsymbol{\theta}})\}. \quad (5)$$

2.2 Space-Time Covariance Matrix Functions

Research in developing geostatistical space-time covariance function models is intensive, given its crucial role in prediction and uncertainty quantification. The space-time covariance function is used to generate a positive definite covariance matrix whose parameters can be calibrated using the log-likelihood with a memory complexity of $O(n^2)$ and computation complexity of $O(n^3)$, where n represents the dimension of the covariance matrix. A popular space-time covariance function proposed in [12] has the form:

$$C(\mathbf{h}, u) = \frac{\sigma^2}{|u|^{2\alpha}/a_t + 1}\mathcal{M}_\nu\left\{\frac{\|\mathbf{h}\|/a_s}{(|u|^{2\alpha}/a_t + 1)^{\beta/2}}\right\}, \quad (6)$$

where $\mathcal{M}_\nu$ is the univariate Matérn correlation function with parameter vector $\boldsymbol{\theta} = (\sigma^2, a_s, \nu, a_t, \alpha, \beta)^\top \in \mathbb{R}^6$, such that $\sigma^2 > 0$ is the variance parameter, $\nu > 0$ and $\alpha \in (0, 1]$ are the smoothness parameters in space and time, respectively, $a_s, a_t > 0$ are the range parameters in space and time, respectively, and $\beta \in [0, 1]$ is the space-time interaction parameter. When $\beta = 0$, the model is classified as a separable model such that it factors into its purely spatial and purely temporal components, which implies independence of the space and time aspects. Nonzero values of β imply that the dependence structure in space relates to some degree with the dependence structure in time. The resulting models when $\beta > 0$ are termed nonseparable. In Figure 1, we show two simulated space-time random fields springing from the nonseparable ($\beta = 1$) and separable ($\beta = 0$) versions of the model in (6). The nonseparable and separable models have the same parameter values except for β. From the figure, one can see that the random fields are identical at $t = 1$, however, they evolve in different ways through time. The model in (6) is considered flexible through supporting both separable and non-separable space-time modeling.

2.3 Particle Swarm Optimization

The PSO algorithm belongs to the class of evolutionary computing algorithms. Inspired by the functionality and structure of living entities from the same taxonomy, the evolutionary computing paradigm has been successfully applied to many scientific fields. Although single living entities have no perception of a high-level goal to achieve, non-organized low-level goals achieved through different entities lead to a global solution for a complex problem. There are several examples of applying the evolutionary computing paradigm to solve scientific challenges such as genetic algorithms [33], simulated annealing [9], differential evolution [24, 28], and PSO [17, 27]. PSO is a heuristic global optimization method that mimics biological swarms' social learning, for instance, a shoal of fish or flock of birds. The search space includes a population of candidate solutions, i.e., a set of candidate parameter vectors. The PSO algorithm considers every single solution as a particle that aims to move with a certain speed to scan the global minimum solution's search space. The particle is associated with a velocity vector that directs the next step of the particle-based on the current position and the direction of the best particle position. The PSO algorithm is conceptually simple and compatible with parallelization. This makes PSO a prevalent choice for scientific applications. However, some limitations should be emphasized, such as local optima convergence and stagnation, where swarms stagnate before reaching the global minimum and remain highly diverse. Gaussian geostatistical space-time modeling involves an optimization process to solve a nonconvex estimation problem by maximizing the log-likelihood function. Herein, the PSO algorithm is applied to solve the optimization process in parallel by relying on an existing parallel implementation of the PSO algorithm, i.e., PPSO [37]. This implementation incorporates the pattern search algorithm [19] into the PSO algorithm to generate a more aggressive search method that can remove the limitations of the original algorithm.

3 RELATED WORK

Although space-time models are constantly being developed, research on scaling these models to handle large volumes of space-time data is lagging behind. Large-scale Gaussian geostatistical modeling has been attempted in the past but mostly in the purely spatial setting, i.e., not including the temporal dimension. In [1], a unified exascale geospatial statistics software was proposed. The software is supported with advanced linear algebra solvers and dynamic runtime systems to enable high-performance spatial data processing. The parallelization is performed in the MLE process in the purely spatial context. Because of the added temporal dimension, extending these to space-time is not trivial. A few existing space-time implementations include [39] using OpenCL language with a separable space-time covariance function. In [8], daily pollen levels were reconstructed using the space-time inverse distance weighted (ST-IDW) interpolation approach parallelized on a shared-memory system, i.e., no interactions between nodes.

Air pollution prediction has been performed in the literature through various means. Often, aerodynamic analyses are employed, wherein a computational model is initiated to model the pollution concentrations. This process requires intensive computations and an in-depth understanding of physical models. The Community Multi-scale Air Quality (CMAQ) [30], the Nested Air Quality Prediction Modeling System (NAQPMS) [11], and the WRFChem [32] are some examples of these computational models. Although these models are invaluable tools, they have several drawbacks. For instance, they require defining many domain-specific parameters that only domain scientists can obtain and understand. Statistical methods, which thrive in finding associations and correlations within the dataset, have an advantage compared to the physics-based modeling approaches because practitioners need not first become experts about the process being modeled [18]. Several statistical tools have been used in literature to perform weather prediction with high accuracy [2, 31, 35]. Several works in literature use statistical methods to perform predictions on the air pollution problem [20, 23, 26]. This work provides a framework upon which Gaussian geostatistical space-time models can perform large-scale air pollution predictions. In this work, we aim at providing a high-performance implementation of the space-time model in (6). To our knowledge, this is the first high-performance implementation of this model on distributed-memory systems.

4 PARALLEL MODELING FRAMEWORK

This section provides a detailed description of the proposed parallel framework through a two-level parallelization technique. At the inner level, a dynamic runtime system orchestrates tasks from the parallel linear solvers required during the MLE operation. At the outer level, the PPSO algorithm is used to distribute the optimization process between a set of MPI sub-communicators.

4.1 Parallel Likelihood Function Estimation

High-performance geostatistics modeling software has been proposed in [1] with three main components: a synthetic data generator, an MLE modeling tool, and a predictor. This software mainly relies on `Chameleon` state-of-the-art linear algebra library [6] and `StarPU` dynamic runtime system [5]. Through a set of task-based linear solvers, high-level geostatistical operations were implemented, mainly synthetic spatial data generation, log-likelihood maximization, and spatial prediction functions. The underlying linear solvers are supported by tile-based algorithms, where the covariance matrix is split into a set of small tiles and operate directly on these tiles [3]. Compared to the block-based algorithms, which were widely used in parallel linear algebra software, tile-based algorithms enable a fine-grained look-ahead mechanism to the trailing submatrix and relaxes artifactual loop-ordered and subroutine boundary synchronization points encountered.

Task-based linear solvers rely on dependence analyses derived directly from the serial code and represented in Directed Acyclic Graphs (DAGs). Each DAG defines the tasks through nodes and the data dependencies through arrows. An example of a 4×4 matrix Cholesky factorization DAG is shown in Figure 2 with four tile-based operations: POTRF, TRSM, SYRK, and GEMM. DAGs

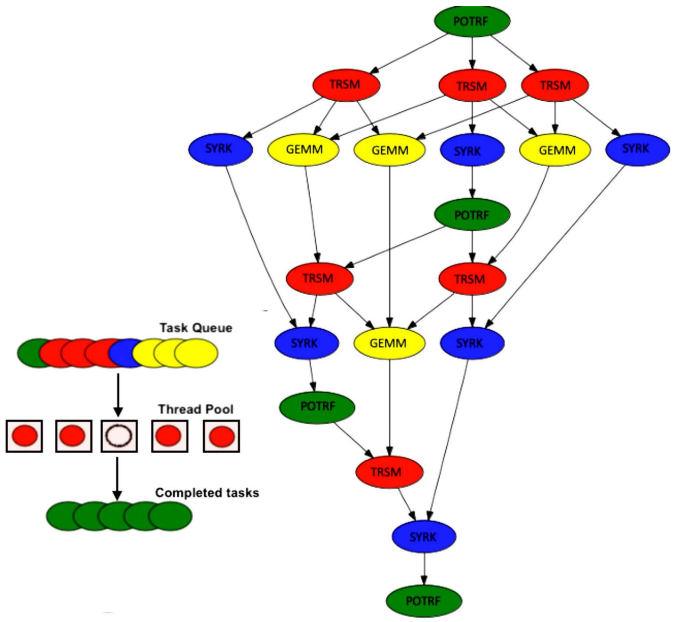
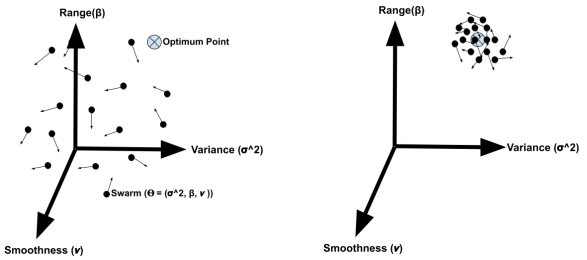


Figure 2: Cholesky factorization 4×4 DAG with four operations: positive-definite triangular Cholesky factorization (POTRF), triangular solve matrix (TRSM), symmetric rank-K update (SYRK), and matrix-matrix multiply (GEMM).

can be sequentially coded and passed to the dynamic runtime system to enable fine-grain execution on the underlying hardware architecture. In [1], `StarPU` dynamic runtime system [4] is used to support the tile-based linear solvers on broad hardware architectures (shared-memory, GPUs, and distributed-memory systems). `StarPU` can directly manage sequential coded DAGs to schedule tasks to the available hardware resources. The `StarPU` internal scheduler ensures the utilization of the resources and the tasks' ordering to prevent possible access violations. `StarPU` provides a high level of abstraction to improve user productivity and creativity. In the proposed implementation, we adopt task-based parallelism in the MLE operation through the `Chameleon` parallel linear algebra library and `StarPU` runtime system.

4.2 Proposed Parallel Optimization Strategy

The MPI API is the predominant programming method in developing HPC applications on large-scale systems [25]. It relies on exchanging data between distributed-memory processing units through message passing. Distributed applications usually require collective operations that involve a group of processes communicating in an isolated context through what is called the MPI communicator. The MPI default communicator is *MPI_COMM_WORLD*, which involves all the initiated MPI processes defined by the user. This default communicator can be split into x sub-communicators where processes in each sub-communicator (x_i) can exchange data through the collective operations associated with this MPI sub-communicator. The *MPI_Comm_split* MPI function is a common way to partition the default communicator into disjoint subgroups



(a) Iteration 0 **(b) Iteration** N

Figure 3: PPSO-based MLE optimization of a univariate purely spatial covariance function.

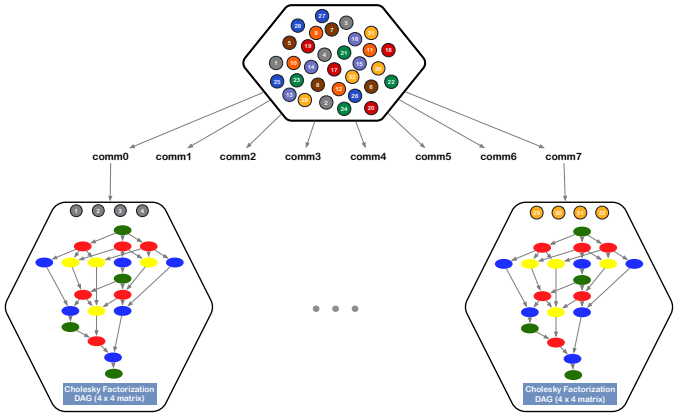


Figure 4: Testcase using 32 nodes and 8 MPI sub-communicators. Each sub-communicator includes 4 nodes that estimate the log-likelihood function with a certain set of parameters in parallel using the `StarPU` runtime system.

associated with different sub-communicators. Each group of processes has one value of color so that each sub-communicator contains all processes of the same color. In this work, we exploit the MPI communicator/sub-communicators to perform a high-level parallelization, optimize the MLE process, and effectively utilize the underlying hardware resources.

Although the state-of-the-art parallel linear algebra solvers provide a high level of parallelization and scalability on large systems, enough workload should be guaranteed for all processing units, which cannot be satisfied because of the memory limits. Thus, we pick the required number of nodes based on the problem size. This should satisfy the higher performance requirement and avoid memory size restrictions. Further scalability can be achieved by parallelizing the optimization process on a larger number of nodes.

In the log-likelihood function, $\widehat{\boldsymbol{\theta}} = (\widehat{\theta_1}, ..., \widehat{\theta_q})^\top$ represents the parameter vector of the maximum likelihood estimates which maximizes the log-likelihood function. Finding the maximum value of the log-likelihood function requires several optimization iterations over a pool of parameter vectors. In the proposed framework, we rely on the PPSwarm software, a parallel implementation of the PPSO algorithm mentioned in Section 2.3 [37]. The parallel implementation utilizes several particles to scan the search space simultaneously, such that each particle can find the maximum value of the log-likelihood function using a single parameter vector on one MPI node. Thus, a single MPI communicator is used to organize the communication between different nodes, i.e., *MPI_COMM_WORLD*. We incorporate the PPSO algorithm with the runtime-based MLE implementation. Assuming several MPI nodes under the default MPI communicator, we split the default communicator into a set of sub-communicators where each can be used to evaluate a single log-likelihood function solution. In this way, we can evaluate several log-likelihood functions in parallel while keeping the MLE operation's low-level parallelization. Figure 3 shows how different particles move from iteration 0 to iteration N to find the optimum point in the search space. Each particle exclusively evaluates one log-likelihood function.

Figure 4 depicts the two-level node splitting operation. Assuming a set of 32 MPI nodes is used, the *MPI_Comm_split* function can be used to split the default *MPI_COMM_WORLD* communicator into a set of sub-communicators, as shown by the figure. Each sub-communicator includes the nodes that will internally communicate to evaluate a single MLE solution with the aid of the underlying parallel linear algebra library and the dynamic runtime system.

5 PERFORMANCE RESULTS AND ANALYSIS

In this section, the performance of the proposed two-level parallelization technique are assessed using both synthetic and real space-time datasets. We generated synthetic datasets with different spatial and temporal dependence profiles that illustrate various cases that can appear in real data. We also obtained two real PM datasets, each with more than $400K$ space-time locations to demonstrate and validate the prediction accuracy of our implementation. The performance is tested on an Intel-based Cray XC40 system with 6,174 compute nodes, each of which has two 16-core Intel Haswell CPUs at 2.30 GHz and 128 GB of memory. All the experiments were conducted on the whole number of cores with different nodes. We rely on the StarPU runtime system to manage the parallelization at the inner level, where sequential tasks are scheduled dynamically to the available processing units. Inside the node, StarPU uses POSIX Threads (pthreads) to run the tasks on different cores (one thread per core). In addition, StarPU relies on MPI at the node level to enable communication between all processes running on different nodes. We found that using one MPI process per node achieves the best performance on our target system than having more MPI processes per node.

5.1 Qualitative Analysis Using Synthetic Data

In this section, we illustrate the merits of our proposed framework on two fronts. First, we showcase the advantage of using nonseparable over separable models. Second, we test the accuracy of our implementation in recovering the parameters of the nonseparable model (see Eq (6)).

Separable and nonseparable models can both be applied to model the dependence structure of any space-time dataset. In practice, the separable model has been the go-to model of geospatial statisticians for modeling almost any kind of space-time data, separable or nonseparable, since it is easier to fit with fewer parameters and

less computation time than the nonseparable model. However, this simpler model ignores any interaction between space dependence and time dependence, which may be vital for improved prediction accuracy. The nonseparable model addresses this problem by accommodating significant space-time dynamics through the space-time interaction parameter, β. To illustrate the strength of the nonseparable model, we perform some numerical experiments. We simulate space-time realizations at 400 spatial locations $\times$ 100 temporal locations from the model in (6) for each values of $\beta \in \{0.1, 0.5, 1\}$. These datasets are representative of space-time interactions, from weak to strong. To each simulated dataset, we fit both a separable ($\beta = 0$) and a nonseparable model ($\beta \geq 0$) on the training set (90%) using local optimization and perform predictions on the testing set (10%). We perform this simulation, estimation, and prediction for 100 rounds and plot the resulting prediction errors, computed using the mean squared prediction error (MSPE) metric, as boxplots for each scenario. Figure 5 shows the results of the experiments. From the results, one can see that both separable and nonseparable models are at par in prediction performance when the interaction in space-time is weak, i.e., $\beta = 0.1$. The difference between the two models appears when tested in moderate and strong interaction scenarios. As expected, the separable model cannot adequately capture non-negligible space-time interaction. Since a nonseparable model is preferable due to its flexibility in supporting both separable and nonseparable modeling, we check whether the parallel implementation can recover all the parameters of such a model. In this set of experiments, we simulate realizations from the nonseparable model in (6) with varying degrees of space-time dependence. We generate space-time random fields from combinations of weak, moderate, and strong dependence in space and time using different values of a_s and a_t, the parameters responsible for the long-range dependence in space-time, respectively. Figure 6 presents the values of the parameter estimates for every dependence scenario. The median values of the parameter estimates match the true parameter values. This shows that the PPSO can satisfactorily perform in any combination of space-time dependence. The boxplots being wider in scenarios with strong dependence in either space and/or time may be due to having fewer data points with separation distance in space and/or time that are larger than the spatial or temporal effective ranges, which are the distances at which the spatial or temporal correlation drops to approximately 0.05.

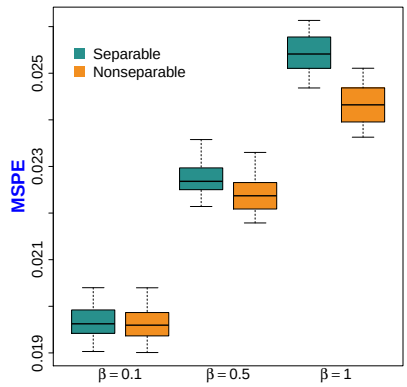


Figure 5: Boxplots of the prediction errors when fitting a separable and nonseparable model on space-time data with varying degrees of space-time interactions. Weak, moderate, and strong space-time interactions are represented by $\beta = 0.1, 0.5,$ and 1, respectively.

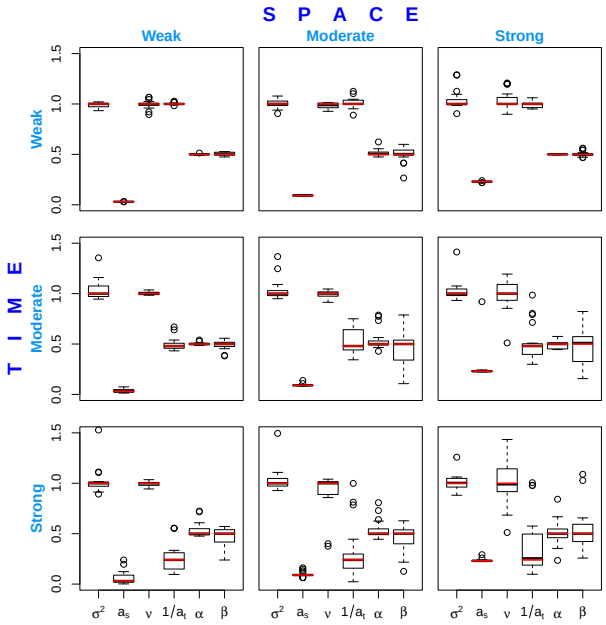


Figure 6: Boxplots of parameter estimates under varying degrees of space-time dependence when $\sigma^2 = 1$, $\nu = 1$, $\alpha = 0.5$, and $\beta = 0.5$. Weak, moderate, and strong dependence in space and time corresponds to $a_s = \{0.03, 0.09, 0.23\}$ and $1/a_t = \{1, 0.48, 0.24\}$, respectively. The true parameters are highlighted in red.

5.2 Qualitative Analysis Using Real PM2.5 Data

In this section, we apply the proposed framework to real space-time datasets downloaded from the NASA Earthdata website. The datasets are Modern-Era Retrospective Analysis for Research and Applications, Version 2 (MERRA-2) reanalysis datasets comprised of hourly PM2.5 measurements. PM2.5 are particulate matters (PMs) with an aerodynamic diameter of 2.5μm or less [36]. The first dataset includes the residuals of the log PM2.5 averaged over 48 hours in Saudi Arabia from 2016 to 2019. Such data transformation ensures that the measurements satisfy the zero-mean, Gaussianity, and stationarity in the space-time second-order structure assumptions; see Section 1, where these assumptions were laid out. The dataset contains 550 spatial locations, each with 730 temporal measurements. The resulting space-time covariance matrix for this dataset is of size $401,500 \times 401,500$. Figure 7 shows the measurements at every spatial location at the first six-time points. We expect some substantial correlation in space from the data as there are recognizable blobs of yellow, red, and blue, indicating that the measurement at any spatial location is similar to its neighbors. We may expect a

Table 1: A summary of the estimated parameters of the nonseparable (NS) and separable (S) models and their corresponding errors (MSPE) and prediction uncertainty (PU) for the Saudi Arabia and US datasets. MSPE1 and PU1 correspond to Testing Dataset 1, while MSPE2 and PU2 point to Testing Dataset 2. The best model reports the lower MSPE and PU.

	Model	$\hat{\sigma}^2$	$\hat{a}_s$	$\hat{\nu}$	$\hat{a}_t$	$\hat{\alpha}$	$\hat{\beta}$	MSPE1 / PU1	MSPE2 / PU2
Saudi Arabia	NS	1.2942	1.3461	2.1568	1.1284	0.1401	0.7548	0.00179877 / 76.91	1.08391 / 875.85
	S	2.6194	1.2737	2.1544	2.0466	0.0308	0	0.00180 / 78.63	1.14705 / 1066.00
US	NS	0.4757	1.3383	1.1266	6.7722	0.7262	0.1451	0.00280/ 155.27	0.05822 / 322.09
	S	2.1237	1.5409	1.4709	7.9975	0.4867	0	0.00318 / 134.52	0.06386 / 1118.20

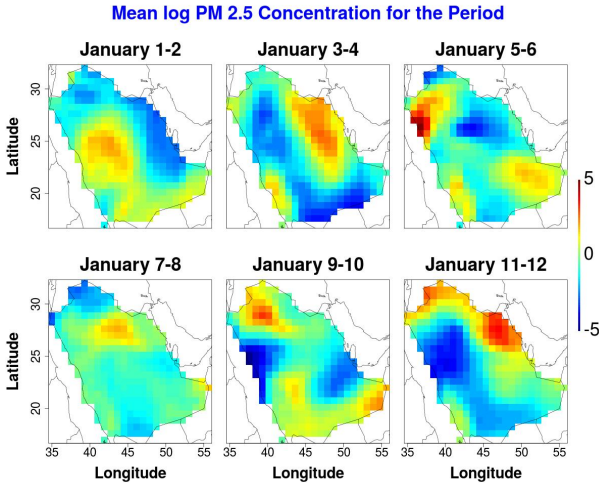


Figure 7: Visualization of the log PM2.5 dataset after space-time mean removal at the first six time points in 2016 over Saudi Arabia.

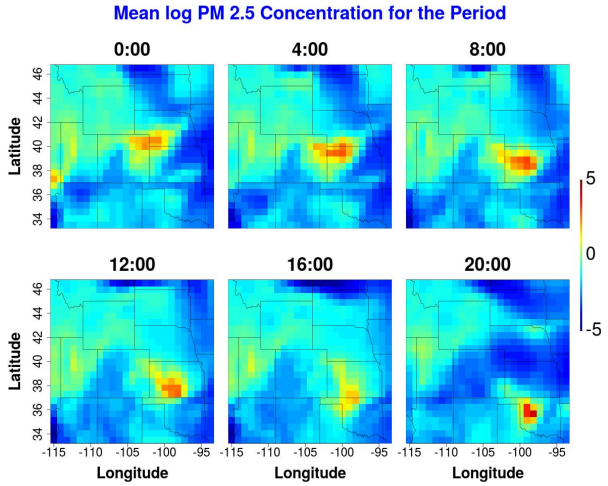


Figure 8: Visualization of the log PM2.5 dataset after space-time mean removal at four hour intervals on January 1, 2016 over the Midwest US.

weak correlation in time as the spatial images are substantially different from one-time point to the next. The second dataset contains the residuals of the hourly log PM2.5 in the Midwest US, where the terrains are relatively flat. Over this certain region, the hourly sampled log PM2.5 already satisfies the stationarity assumption, hence, no averaging is performed. This other dataset contains 945 spatial locations, each with 500 temporal measurements, with a resulting space-time covariance matrix of size $472,500 \times 472,500$. Figure 8 shows the measurements of the second dataset at every spatial location at four-hour intervals. Similarly, moderate to strong correlation in space can be seen. However, unlike the first dataset, a strong correlation in time can be hypothesized as the spatial image structure from the first hour tends to persist even until after 20 hours. Furthermore, Figure 8 illustrates a unique phenomenon called the transport phenomenon, where objects such as PM2.5 particles get transported from one place to another by some media such as the wind. Such behavior can be detected by following the red blob at 0:00 located over Colorado and Nebraska. At 20:00, the red blob has traveled to Oklahoma. This transport phenomenon and many other environmental phenomena are more appropriately modeled by nonseparable space-time models [12, 34].

To validate our method, we prepare two datasets for estimation and testing. Testing Dataset 1 comprises of measurements at randomly selected space-time locations within $t = 1$ to $t = 730$ for Saudi Arabia and within $t = 1$ to $t = 500$ for the US. On the other hand, Testing Dataset 2 comprises of measurements at all spatial locations two-time steps in the future, i.e., at $t = 731$ and $t = 732$ for Saudi Arabia and at $t = 501$ and $t = 502$ for the US. We obtain the estimated parameters, and predict the measurements (Equation 4) and compute the prediction errors measured by the MSPE and the associated prediction uncertainty (PU) (Equation 5). The results are reported in Table 1. The parameter estimates in Table 1 imply several properties regarding the two space-time random fields. First, the nonseparability parameter, β, for Saudi Arabia has an estimated value of 0.7548. This is higher than the estimated β value of 0.1451 for the US. Therefore, the space-time interaction for log PM2.5 is much stronger in Saudi Arabia than in the US. Second, the range parameter in time, a_t, has an estimated value that is higher in the US than that in Saudi Arabia. This suggests that log PM2.5 values at time points that are far apart remain strongly correlated. Based on the parameter values, for the Saudi Arabia case, the correlation drops to 0.05 after 8-time points, while the correlation drops to 0.05 after 150-time points for the US one. Third, some parameters

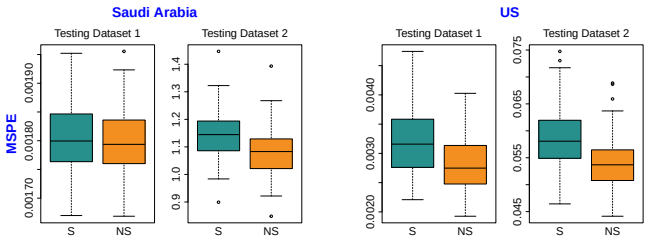


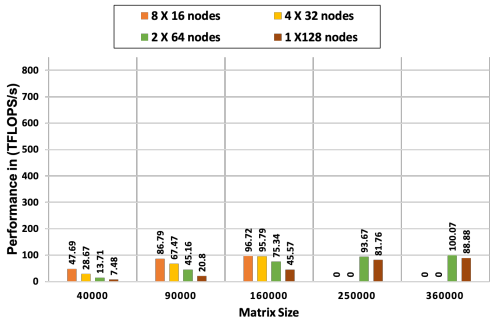
Figure 9: Boxplots of the prediction errors for the separable (S) and nonseparable (NS) models in the real data pseudo cross-validation study.

in the separable model differ a lot from their counterparts in the nonseparable model. This can be expected as the parameters in the separable model need to compensate for the lack of the β parameter. In particular, the variance parameter, σ^2, is artificially inflated and is the most impacted parameter. Moreover, when we compute the sample variances of the training dataset for Saudi Arabia and the US, we obtain the values 0.9979 and 0.8050, respectively. The variance parameter estimates under the nonseparable model are closer to these values obtained from the real data. This is one disadvantage of the separable model, i.e., we obtain erroneous parameter estimates that are not supported by the real data. Fourth, the errors obtained in Testing Dataset 1 are smaller than those obtained in Testing Dataset 2. This is expected as predicting on Testing Dataset 1 involves filling the missing values at some locations in space for each time point. On the other hand, predicting on Testing Dataset 2 entails forecasting forward in time where all values on the map are missing for several time points ahead. Prediction on datasets such as Testing Dataset 2 is much more common in practice, wherein full maps of measurements in the following few time points are desired. Lastly, the separable model returns higher prediction errors than the nonseparable model. In particular, the percentages of improvements of the nonseparable over the separable model for MSPE in Testing Dataset 2 is 6% in Saudi Arabia and 9% in the US, and the prediction uncertainty (PU) is 18% in Saudi Arabia and 71% in the US.

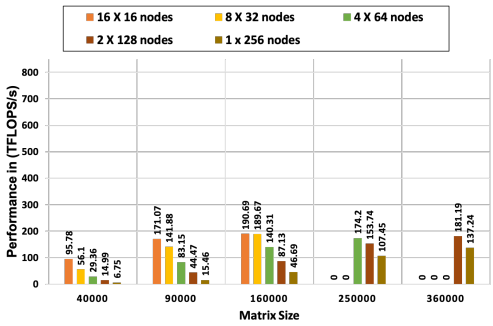
The difference in MSPEs between the nonseparable and separable models may not be huge because the values are small. Hence, to further show that the nonseparable model is superior, we perform a pseudo cross-validation [29] study wherein for 100 rounds, we take a subset of each of the testing datasets and compute the resulting MSPE. We collect the MSPE values and illustrate the distribution as boxplots in Figure 9. The boxplots have shown that indeed the nonseparable model returns lower errors in both weak correlation (Saudi Arabia) and strong correlation (US) scenarios. These real data results mirror the synthetic data experiments results in Figure 5.

5.3 Performance Evaluation

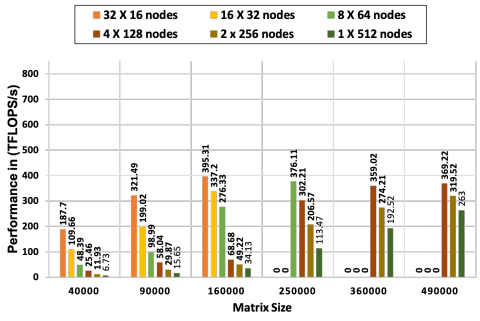
We aim in this section to evaluate the performance of the proposed two-level parallel implementation. Parallelization of the MLE operation through a set of MPI sub-communicators requires determining the number of nodes in each MPI sub-communicator that satisfies a decent performance (execution time and FLOPS/s). Assuming a single MPI process per node, the workload is distributed to run through



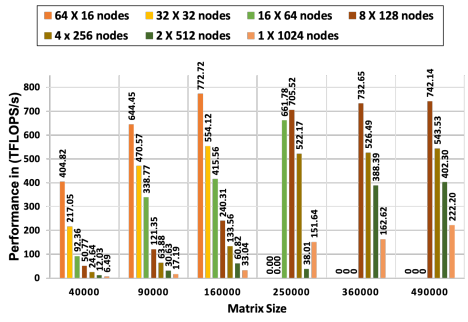
(a) 128 nodes.



(b) 256 nodes.



(c) 512 nodes.



(d) 1024 nodes.

Figure 10: Performance of a single MLE optimization step using a different number of nodes on Shaheen-II Cray XC40 Supercomputer. The legends show how many MPI sub-communicators x are used and how many y nodes exist in each MPI sub-communicator.

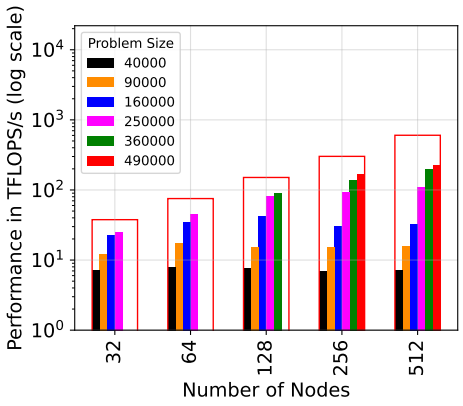
a set of MPI sub-communicators. Each MPI sub-communicator has some nodes less than or equal to the total number of target nodes. We use $x \times y$ notation to define a specific workload distribution where x represents the number of MPI sub-communicators and y represents the number of nodes in each sub-communicator. Herein, we evaluate these combinations for a set of given nodes, i.e., 32, 64, 128, 256, 512, and 1024 on KAUST Shaheen-II Cray XC40 system. We assume a load-balanced workload distribution since all computations are densely performed (full-double precision).

Figure 10 shows the performance of different combinations, i.e., $x \times y$ nodes using different matrix sizes. The missing data come with the memory limitation of using a small number of nodes per MPI sub-communicator. For the different number of nodes, the same remarks hold. First, fewer nodes per MPI sub-communicator (i.e., distributing the workload through more MPI sub-communicators) is the main option to achieve decent performance. Of course, this is restricted by the size of the problem. The best performance can be obtained by the smallest number of nodes per sub-communicator that can handle a given problem size. Second, the performance improvement can be clearly observed in all the subfigures with small matrix sizes and a small number of nodes compared to a large number of nodes per sub-communicators since the workload is small and cannot saturate a large number of nodes. For instance, in Figure 10a, using 128 nodes, the performance improvement of using eight MPI sub-communicators compared to the single communicator with all the 128 nodes is about 6.38X with a problem size of 40K. However, in the case of 160K, only 2.12X performance improvement has been achieved. Another example can be captured from Figure 10c, using 512 nodes, the achieved performance improvements are 27.89X and 11.58X with 40K and 160K problem sizes, respectively. Third, the proposed framework shows higher performance improvements with more nodes. For instance, using 160K problem size, the achieved performance improvements are 2.12X, 4.08X, 11.58X, and 23.39X using 128, 256, 512, and 1024 nodes, respectively. We summarize the performance gain of applying the two-level parallel MLE implementation compared to the one-level parallel implementation in Figures 11a and 11b. We use both performance (FLOPS/s) and time-to-solution for the comparison. In the two-level parallel MLE implementation, the number of MPI sub-communicators has been tuned for each number of nodes and problem size to achieve the highest performance.

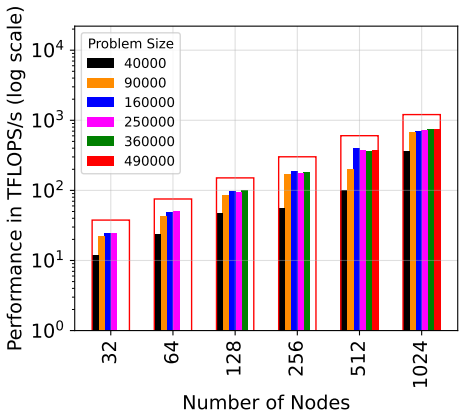
In Figure 11, the x- and y-axes show the number of nodes and the performance (TFLOPS/s) in log-scale. Six different matrix sizes are used for the evaluation, i.e., 40K, 90K, 160K, 250K, 360K, and 490K. Figure 11a reports the one-level implementation performance where only the parallel linear solvers are applied to the log-likelihood function. The empty red box shows the peak performance (Rpeak) that a certain number of nodes can theoretically achieve.

With smaller workloads, the performance degrades with a larger number of nodes when there is not enough work to keep all the nodes busy. The missing data come with the lack of saturating the target number of nodes). Moreover, compared to the peak performance of each number of nodes (in the red curve), the one-level parallelization can achieve up to 67%, 59%, 60%, 56%, 47%, and 28% of Rpeak using 32, 64, 128, 256, 512, and 1024, respectively.

Figure 11b shows performance using the two-level parallel implementation. The figure confirms the scalability of the proposed



(a) One-level parallelization.



(b) Two-level parallelization.

Figure 11: Performance of one MLE optimization step using single and n MPI communicators on Shaheen-II Cray XC40 Supercomputers. In (b) x MPI sub-communicators is used where x is tuned for performance.

implementation with the different number of nodes. We report the best performance of different numbers nodes and problem sizes with a tuned x value, i.e., the number of sub-communicators. The achieved performance compared to the peak performance can reach up to 65%, 66%, 66%, 60%, 59%, and 58% using 32, 64, 128, 256, 512, and 1024, respectively. Compared to Figure 11a, the proposed implementation satisfies stable performance with a different number of nodes and with different problem sizes.

Figure 12 highlights the gained performance in using the proposed two-level parallelization compared to the one-level parallelization using 1024 nodes on Shaheen-II Cray XC40 system. The figure shows that the proposed framework can achieve 3.3X more TFLOPS/s than the baseline framework using a 490K dataset. Smaller matrix sizes can achieve higher performance increases, but we think it might be overkill to highlight this speedup since the scalability of the one-level parallelization degraded so much with small problem size and a large number of nodes. Of course, performance analysis addresses processing capability; however, time-to-solution is

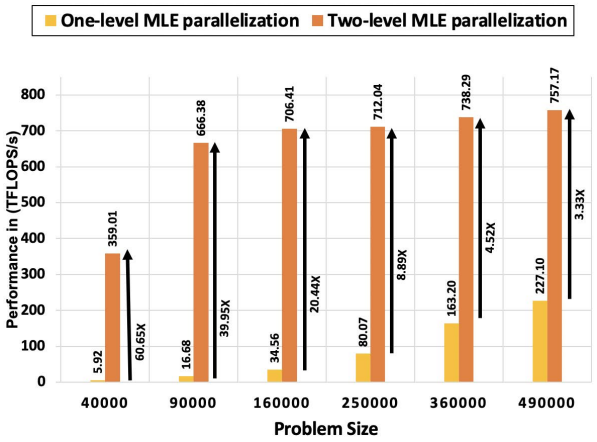
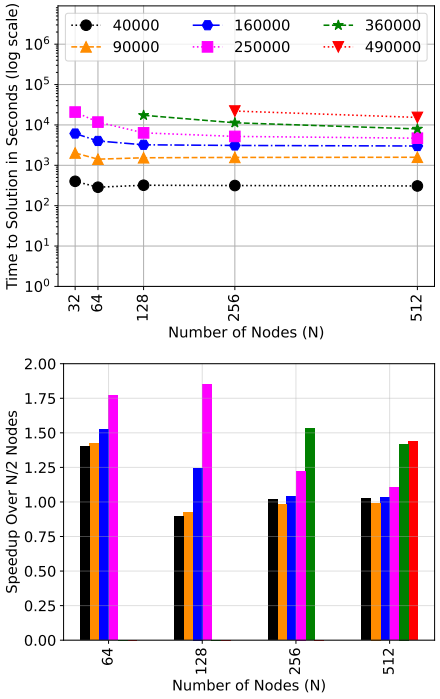


Figure 12: One-level versus two-level MLE parallelization performance using 1024 nodes on Shaheen-II Cray XC40 system.

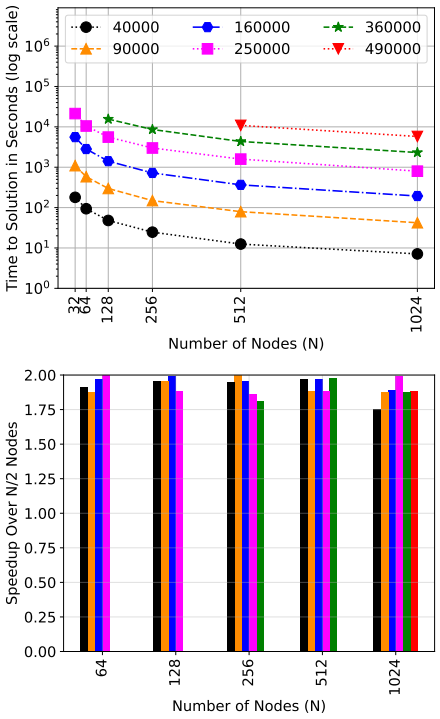
also essential to assess the effectiveness of the proposed method. Figure 13 shows the time-to-solution, where the number of optimization iterations is fixed to 100. When one-level parallelism is used (Figure 13a), smaller workloads take the same execution time even when the number of nodes is increased. Thus, at a certain point, increasing the number of nodes will not improve the time-to-solution anymore. For instance, with a 360K problem size, using 1024 nodes instead of 512 nodes will not accelerate the execution time. On the contrary, the proposed implementation keeps the execution time speedup gains with more nodes. For instance, with a 360K problem size, the total execution time of the MLE operation is 4343.42 and 2311.48 seconds with 256 and 512 nodes, respectively, with a 1.87X speedup. Figure 13 also summarizes the achieved speedup by the one-level and the two-level parallel implementations when doubling the number of nodes. As shown, the proposed two-level parallel implementation always achieves speedup close to theoretical speedup, i.e., N nodes can achieve $2X$ speedup compared to $N/2$ nodes.

6 CONCLUSION

We proposed a parallel implementation of geostatistical space-time modeling that can predict air pollution using observations in a specific space-time domain, illustrating the importance of relaxing the assumption of independence of space and time. The proposed two-level framework relies on MPI sub-communicators at the upper level to perform independent log-likelihood function evaluation with different sets of parameters. These independent computations result from applying parallel optimization through the PPSO algorithm, where each swarm performs a single log-likelihood estimation. A task-based parallel technique is used at the inner level to perform linear solver operations on a given set of nodes representing a single MPI sub-communicator. StarPU runtime system have been used to schedule the tasks to target processing units. The performance of the proposed implementation has been assessed using synthetic and real datasets from two different geographical



(a) One-level parallelization.



(b) Two-level parallelization.

Figure 13: Time-to-solution of full MLE operation using 100 optimization iterations on Shaheen-II Cray XC40 system. In (b), we tune x MPI sub-communicators for performance.

regions and the importance of using a space-time model that incorporates the interaction between the spatial and temporal domains was demonstrated. We achieved high prediction accuracy on pollution data with 490,000 locations with up to 757 TFLOPS/s using 1024 nodes on the KAUST Shaheen-II Cray XC40 system (around 63% of the theoretical peak). Future work would involve applying tile low-rank and mixed-precision approximations to accelerate further the modeling process.

ACKNOWLEDGEMENT

We would like to acknowledge the Office of Sponsored Research (OSR) at King Abdullah University of Science and Technology (KAUST) to fund and support this work. We would also like to acknowledge Intel for support in the form of an Intel Parallel Computing Center award Cray for the support provided during the Center of Excellence award to the Extreme Computing Research Center at KAUST. This work and research used resources from the Extreme Computing Research Center (ECRC) and the KAUST Supercomputing Laboratory at KAUST, including Cray XC40, KAUST Shaheen-II supercomputer.

REFERENCES

[1] Sameh Abdulah, Hatem Ltaief, Ying Sun, Marc G Genton, and David E Keyes. 2018. ExaGeoStat: A high performance unified software for geostatistics on manycore systems. *IEEE Transactions on Parallel and Distributed Systems* 29, 12 (2018), 2771–2784.

[2] Sameh Abdulah, Hatem Ltaief, Ying Sun, Marc G Genton, and David E Keyes. 2019. Geostatistical modeling and prediction using mixed precision tile cholesky factorization. In *2019 IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC)*. IEEE, 152–162.

[3] Emmanuel Agullo, Jim Demmel, Jack Dongarra, Bilel Hadri, Jakub Kurzak, Julien Langou, Hatem Ltaief, Piotr Luszczek, and Stanimire Tomov. 2009. Numerical linear algebra on emerging architectures: The PLASMA and MAGMA projects. In *Journal of Physics: Conference Series*, Vol. 180. IOP Publishing, 012037.

[4] Cédric Augonnet, Samuel Thibault, Raymond Namyst, and Pierre-André Wacrenier. 2009. StarPU: A unified platform for task scheduling on heterogeneous multicore architectures. In *European Conference on Parallel Processing*. Springer, 863–874.

[5] Cédric Augonnet, Samuel Thibault, Raymond Namyst, and Pierre-André Wacrenier. 2011. StarPU: a unified platform for task scheduling on heterogeneous multicore architectures. *Concurrency and Computation: Practice and Experience* 23, 2 (2011), 187–198.

[6] CHAMELEON 2021. The Chameleon Project: A dense linear algebra software for heterogeneous architectures. Available at https://project.inria.fr/chameleon/.

[7] Wanfang Chen, Marc G Genton, and Ying Sun. 2021. Space-time covariance structures and models. *Annual Review of Statistics and Its Application* 8 (2021), 191–215.

[8] Michael R Desjardins, Alexander Hohl, Adam Griffith, and Eric Delmelle. 2019. A space–time parallel framework for fine-scale visualization of pollen levels across the Eastern United States. *Cartography and Geographic Information Science* 46, 5 (2019), 428–440.

[9] Kathryn Anne Dowsland and Jonathan Thompson. 2012. Simulated annealing. *Handbook of Natural Computing* (2012), 1623–1655.

[10] Goran Gašparac, Amela Jeričević, Prashant Kumar, and Branko Grisogono. 2020. Regional-scale modelling for the assessment of atmospheric particulate matter concentrations at rural background locations in Europe. *Atmospheric Chemistry and Physics* 20, 11 (2020), 6395–6415.

[11] Baozhu Ge, Zifa Wang, Xiaobin Xu, Jianbin Wu, Xiaolan Yu, and Jian Li. 2014. Wet deposition of acidifying substances in different regions of China and the rest of East Asia: Modeling with updated NAQPMS. *Environmental Pollution* 187 (2014), 10–21.

[12] Tilmann Gneiting. 2002. Nonseparable, stationary covariance functions for space–time data. *J. Amer. Statist. Assoc.* 97, 458 (2002), 590–600.

[13] Tilmann Gneiting, Marc G Genton, and Peter Guttorp. 2007. Geostatistical space-time models, stationarity, separability, and full symmetry. *Finkenstaedt, B., Held, L. and Isham, V. (eds.), Statistics of Spatio-Temporal Systems, Chapman & Hall/CRC Press, Monographs in Statistics and Applied Probability* 107 (2007), 151–175.

[14] Yong H Huang, James E Saiers, Judson W Harvey, Gregory B Noe, and Steven Mylon. 2008. Advection, dispersion, and filtration of fine particles within emergent vegetation of the Florida Everglades. *Water Resources Research* 44, 4 (2008).

[15] Konstantinos E Kakosimos, Marc J Assael, John S Lioumbas, and Anthimos S Spiridis. 2011. Atmospheric dispersion modelling of the fugitive particulate matter from overburden dumps with numerical and integral models. *Atmospheric Pollution Research* 2, 1 (2011), 24–33.

[16] George Kallos, Marina Astitha, Petros Katsafados, and Chris Spyrou. 2007. Long-range transport of anthropogenically and naturally produced particulate matter in the Mediterranean and North Atlantic: Current state of knowledge. *Journal of Applied Meteorology and Climatology* 46, 8 (2007), 1230–1251.

[17] James Kennedy and Russell Eberhart. 1995. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, Vol. 4. IEEE, 1942–1948.

[18] Byeong Soo Kim, Bong Gu Kang, Seon Han Choi, and Tag Gon Kim. 2017. Data modeling versus simulation modeling in the big data era: case study of a greenhouse control system. *Simulation* 93, 7 (2017), 579–594.

[19] Robert Michael Lewis and Virginia Torczon. 2002. A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization* 12, 4 (2002), 1075–1089.

[20] Bing-Chun Liu, Arihant Binaykia, Pei-Chann Chang, Manoj Kumar Tiwari, and Cheng-Chin Tsao. 2017. Urban air quality forecasting based on multi-dimensional collaborative support vector regression (svr): A case study of Beijing-Tianjin-Shijiazhuang. *PLOS One* 12, 7 (2017), e0179763.

[21] Matthew Loxham, Donna E Davies, and Stephen T Holgate. 2019. The health effects of fine particulate air pollution.

[22] Michael C McCarthy, Douglas S Eisinger, Hilary R Hafner, Lyle R Chinkin, Paul T Roberts, Kevin N Black, Nigel N Clark, Peter H McMurry, and Arthur M Winer. 2006. Particulate matter: a strategic vision for transportation-related research. *Environmental Science & Technology* 40, 18 (2006), 5593–5599.

[23] J Murillo-Escobar, JP Sepulveda-Suescun, MA Correa, and D Orrego-Metaute. 2019. Forecasting concentrations of air pollutants using support vector regression improved with particle swarm optimization: Case study in Aburrá Valley, Colombia. *Urban Climate* 29 (2019), 100473.

[24] Karol R Opara and Jarosław Arabas. 2019. Differential Evolution: A survey of theoretical analyses. *Swarm and Evolutionary Computation* 44 (2019), 546–558.

[25] Ashwin Pajankar. 2017. Message passing interface. In *Raspberry Pi Supercomputing and Scientific Programming*. Springer, 61–65.

[26] Eleni Petrakou and Iasonas Topsis Giotis. 2020. Planetary statistics and forecasting for solar flares. *arXiv preprint arXiv:2006.10694* (2020).

[27] Riccardo Poli, James Kennedy, and Tim Blackwell. 2007. Particle swarm optimization. *Swarm Intelligence* 1, 1 (2007), 33–57.

[28] Kenneth V Price. 2013. Differential evolution. In *Handbook of optimization*. Springer, 187–214.

[29] Payam Refaeilzadeh, Lei Tang, and Huan Liu. 2009. Cross-validation. *Encyclopedia of Database Systems* 5 (2009), 532–538.

[30] Allison M Ring, Timothy P Canty, Daniel C Anderson, Timothy P Vinciguerra, Hao He, Daniel L Goldberg, Sheryl H Ehrman, Russell R Dickerson, and Ross J Salawitch. 2018. Evaluating commercial marine emissions and their role in air quality policy using observations and the CMAQ model. *Atmospheric Environment* 173 (2018), 96–107.

[31] Mary Lai Salvana, Sameh Abdulah, Huang Huang, Hatem Ltaief, Ying Sun, Marc Genton, and David Keyes. 2021. High Performance Multivariate Geospatial Statistics on Manycore Systems. *IEEE Transactions on Parallel and Distributed Systems* 32 (2021), 2719–2733.

[32] Pierre Sicard, Paola Crippa, Alessandra De Marco, Stefano Castruccio, Paolo Giani, Juan Cuesta, Elena Paoletti, Zhaozhong Feng, and Alessandro Anav. 2021. High spatial resolution WRF-Chem model over Asia: physics and chemistry evaluation. *Atmospheric Environment* 244 (2021), 118004.

[33] SN Sivanandam and SN Deepa. 2008. Genetic algorithms. In *Introduction to genetic algorithms*. Springer, 15–37.

[34] Michael L Stein. 1999. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media.

[35] Zhongda Tian, Gang Wang, and Yi Ren. 2020. Short-term wind speed forecasting based on autoregressive moving average with echo state network compensation. *Wind Engineering* 44, 2 (2020), 152–167.

[36] W Gene Tucker. 2000. An overview of PM2.5 sources and control strategies. *Fuel Processing Technology* 65 (2000), 379–392.

[37] A Ismael F Vaz and Luís N Vicente. 2007. A particle swarm pattern search method for bound constrained global optimization. *Journal of Global Optimization* 39, 2 (2007), 197–219.

[38] Jacqueline Whalley and Sara Zandi. 2016. Particulate matter sampling techniques and data modelling methods. In *Air Quality-Measurement and Modeling*. INTECH, 10.

[39] Yueheng Zhang, Xinqi Zheng, Zhenhua Wang, Gang Ai, and Qing Huang. 2018. Implementation of a parallel GPU-based space-time kriging framework. *ISPRS International Journal of Geo-Information* 7, 5 (2018), 193.

[40] Chi Zhou, HB Gao, Liang Gao, and WG Zhang. 2003. Particle Swarm Optimization (PSO) Algorithm [J]. *Application Research of Computers* 12 (2003), 7–11.